I CLAIM:

Sub A1

1. A method for processing an exception in an emulator program running on a digital computer having a memory and under control of an operating system, the emulator program emulating execution of a user program constructed for execution on a legacy platform, the method comprising the steps of:

receiving an exception from the operating system;

determining whether the received exception was caused by the emulator program itself or by the user program; and

if the exception was caused by the emulator program, handling the exception internally in the emulator program without delivering the exception to the emulated user program.

09035174.030598

2. A method for processing an exception according to claim 1 and further comprising, if the exception was caused by the user program:

identifying the type of exception;

determining whether the identified type of exception is currently blocked by the user program; and

if the identified type of exception is not currently blocked by the user program, delivering notification of the exception to the user program.

Sub C1

3. A method for processing an exception according to claim 2 and further comprising, if the identified type of exception is currently blocked by the user program, withholding delivery of the exception from the user program, and marking the exception as deferred for subsequent processing.

4. A method for processing an exception according to claim 3 wherein said determining whether the identified type of exception is blocked by the user program includes maintaining a virtual exception mask for simulating a user program exception mask as if the user program were running on the legacy platform.

~~5. A method for processing an exception according to claim 4, the digital computer further including an OS exception mask maintained by the operating system, and wherein said maintaining the virtual exception mask includes:~~

~~intercepting a user program system call;~~

~~determining whether the system call would modify the OS exception mask;~~

~~if the system call would not modify the OS exception mask, delivering the system call to the operating system; and~~

~~if the system call would modify the OS exception mask, updating the virtual exception mask in accordance with the system call.~~

~~6. A method for processing an exception according to claim 5 and further comprising, responsive to said updating the virtual exception mask, determining whether any deferred exceptions are now unblocked according to the updated virtual exception mask;~~

~~delivering any unblocked exceptions to the user program; and then, for each such delivered unblocked exception, clearing the indication that the delivered exception had been deferred.~~

[illegible]

~~7. A method for processing an exception according to claim 4 wherein said maintaining the virtual exception mask includes storing the virtual exception mask as a predetermined data structure within the emulator memory space.~~

~~8. A method for processing an exception according to claim 3 further comprising creating and maintaining a status mask for indicating a status as deferred or not deferred for each one of at least one predetermined type of exception; and wherein said marking the exception as deferred includes updating the status mask to indicate a status of the exception as deferred.~~

[illegible]

~~9. A method according to claim 8 wherein the status mask is implemented as a predetermined data structure within the emulator memory space.~~

Sub A2

10. A method for processing an exception according to claim 1 and further comprising, if the exception is determined to have been caused by the user program, and if the exception is not currently blocked by the user program,

determining whether the exception is synchronous or asynchronous; and if the exception is synchronous, delivering the exception to the user program.

11. A method for processing an exception according to claim 10 and further comprising, if the exception is asynchronous, determining whether the exception indicates an interrupted system call; and if not, marking the exception as pending.

12. A method for processing an exception according to claim 11 and further comprising, if the exception is asynchronous, and if the exception indicates an interrupted system call, delivering the exception to the user program.

13. A method of maintaining a virtual exception mask in a digital computer for emulation, the computer having a memory, an operating system and an OS exception mask maintained by the operating system, and the method comprising the steps of:

initializing a virtual exception mask as a data structure stored in the computer memory;

during execution of an emulation in the computer, intercepting a user program system call;

determining whether the system call would modify the OS exception mask;

if the system call would not modify the OS exception mask, delivering the system call to the operating system; and

if the system call would modify the OS exception mask, updating the virtual exception mask in accordance with the intercepted system call.

14. A method according to claim 13 wherein the virtual exception mask includes bits for blocking one or more corresponding synchronous exceptions.

15. A method according to claim 14 wherein the synchronous exceptions include an arithmetic fault and a privilege fault.

16. A method according to claim 13 and further comprising maintaining a status mask as a data structure stored in the computer memory for maintaining an indication of a deferred exception.

17. A digital computer having a memory and including:

a legacy user program stored in the memory and comprising a predetermined series of user program instructions, the legacy user program having been constructed for execution on a predetermined legacy platform other than the said digital computer,

an emulator program stored in the memory for executing the user program instructions by interpreting the user program instructions so as to form corresponding new instructions executable in the said digital computer; and

a virtual exception mask stored in the memory and maintainable by the emulation program for handling exceptions encountered during execution of the said new instructions so as to simulate exception behavior of the legacy user program on the legacy platform.

18. A digital computer according to claim 17 and further comprising means for determining whether an exception encountered during execution of the said new instructions was caused by the said emulator program or one of the said new instructions.

19. A digital computer according to claim 18 and further comprising means for intercepting an exception encountered during execution of the said new instructions in accordance with the virtual exception mask.

20. A digital computer according to claim 19 wherein the digital computer

comprises a VLIW processor architecture and the legacy platform comprises a RISC processor architecture.

Add A3

****

09519350 . 051500